GEOMETRICAL AND TOPOLOGICAL ISSUES IN OCTREE BASED AUTOMATIC MESHING

Mukul Saxena and Renato Perucchio*
Production Automation Project
and Department of Mechanical Engineering
University of Rochester
Rochester, N.Y. 14627, U.S.A.

SUMMARY

Finite element meshes derived automatically from solid models through recursive spatial subdivision schemes (octrees) can be made to inherit the hierarchical structure and the spatial addressability intrinsic to the underlying grid. These two properties, together with the geometric regularity that can also be built into the mesh, make octree based meshes ideally suited for efficient analysis and self-adaptive remeshing and reanalysis. This paper discusses the element decomposition of the octal cells that intersect the boundary of the domain. The problem, central to octree based meshing, is solved by combining template mapping and element extraction into a procedure that utilizes both constructive solid geometry and boundary representation techniques. Boundary cells that are not intersected by the edge of the domain boundary are easily mapped to predefined element topology. Cells containing edges (and vertices) are first transformed into a planar polyhedron and then triangulated via element extractors. This paper also analyzes the modelling environments required for the derivation of planar polyhedra and for element extraction.

1 INTRODUCTION

In this paper, we describe an approach for resolving the geometrical and topological issues that arise when a recursive spatial subdivision scheme (octree) is used to generate automatically a FEM mesh from a solid model. Amongst the various schemes that have been proposed for automatic mesh generation from solid models [WOO84, WÖRD84, CAVE85, SHEP85, YERR85] recursive spatial subdivision schemes have been found to offer an efficient avenue for automatic mesh generation as well as for self-adaptive remeshing and reanalysis because of two intrinsic properties: hierarchical structure and spatial addressability [KELA86]. To understand the importance of these two properties consider the subdivision rule and the associated tree structure illustrated – for a 2-D example – in Figure 1.1. The recursive subdivision rule can be concisely described as follows: (i) the solid domain is "boxed" and the box is decomposed into quadrants (octants in 3-D); (ii) quadrants are classified with respect to the domain: when a quadrant is totally inside or outside of the object, the decomposition ceases; when a quadrant is neither wholly inside nor outside, it is further subdivided into quadrants; (iii) the process continues until some minimal resolution level is reached. The resulting quaternal or octal tree can be thought of as a hierarchical cataloging structure for data describing particular regions of space. The lowest level of the

* Research Assistant and Assistant Professor, respectively

tree (the "resolution" level) contains the smallest spatial regions and the ordinary finite elements. At higher levels the regions become larger and the finite elements become substructures ("superelements") with associated assembled stiffness matrices. As shown in [KELA86], such a hierarchical organization is ideally suited for self-adaptive mesh refinement and incremental analysis. Furthermore, if the quadrant or octant cells are numbered systematically, then the index of any cell in the hierarchical tree can be quickly computed from its size and position, and conversely the size and position of a cell can be directly derived from its index. This property, called spatial addressability, permits direct access to pertinent geometrical and analytical data during both global mesh generation and localized mesh refinement.

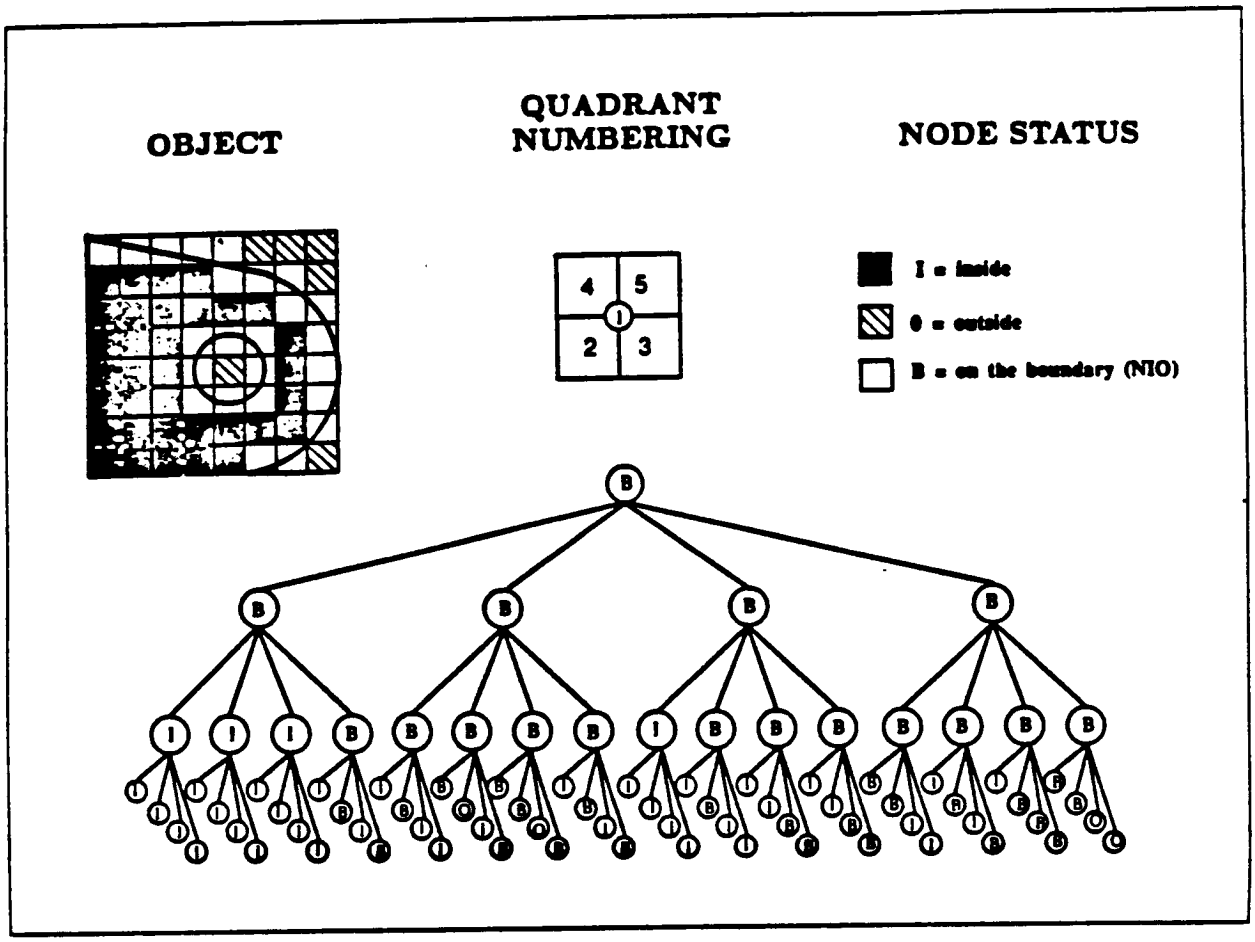

FIGURE 1.1 : Quadrant numbering scheme for a 2-D decomposition.

The octree-based scheme presented here is a direct extension of the work in 2-D meshing and adaptive analysis reported in [KELA86]. The scheme involves two stages of meshing. In the first stage the interior of the domain is meshed with a geometrically regular grid of hexahedral elements that includes all the inside octree cells. In the second stage the mesh is extended to the boundary of the domain by inserting finite element topologies in the 3-manifolds formed by the intersection of the octree cell with the bounding surface of the solid. While for 2-D problems the manifolds are cut quadrants that can be easily decomposed into quadrilateral and triangular elements via template matching, the 3-manifolds associated to boundary intersecting octree cells are far more complex structures that cannot be handled by templates alone. Furthermore, to enforce continuity of the field variable and to maintain the geometrical regularity of the interior mesh, the interface between each 3-manifold and neighboring hexahedral elements must be a square. Since tetrahedral elements are essential for the decomposition of the cells intersected by

the boundary, the interface requirement can be satisfied only by introducing pentahedral ("pyramid") elements to provide the transition between triangular and square faces.

In essence, the crucial problem of octree based meshing is to decompose the cells on the boundary into valid element assemblies while maintaining the hierarchical structure, the spatial addressablity, and the geometrical regularity associated with the underlying octree grid. In the following sections we discuss the meshing scheme in general terms and then we focus on the decomposition of 3-D boundary cells.

2 AN OCTREE BASED MESHING SCHEME

We begin by describing the object in a modelling system based on Constructive Solid Geometry[1] which provides all the basic geometric operators for spatial decomposition and meshing. The idea underlying the octal decomposition scheme is to approximate the object to be meshed with a union of disjoint, variably sized cells (cubes) [JACK80]. However, such an approximation cannot be directly mapped onto a finite element mesh for two crucial reasons: (i) adjoining elements corresponding to octal cells of different size violate the connectivity rules between finite elements, and (ii) the union of orthogonal surfaces that approximates the boundary of the solid contains re-entrant vertices and edges which introduce artificial singularities in the FEM model. We modify the octal decomposition scheme to yield a valid FEM discretization according to the following two-stage strategy.

First stage of Meshing

The object S is enclosed in a "box" and the box is recursively decomposed into octal cells which are classified as being "IN" S, "OUT" of S or neither in nor outside ("NIO"). For IN cells subdivision ceases and the octant is mapped directly on to a finite element substructure. OUT cells are discarded and NIO cells are further subdivided and classified until a pre-specified level of subdivision (the "resolution" level) is reached and no cell contains more than one connected boundary segment of S. IN cells at resolution level are mapped onto finite elements. The collection of IN cells forms the interior octree of the solid.

Figure 2.1 shows the interior octree for a solid part – a bracket modelled in the PADL-2 domain [HART83]. The classification procedure used in this stage of meshing is described in [LEE82].

Second stage of Meshing

During the second stage the interior octree is extended to the boundary of S, bS. This requires associating each of the NIO cells (more precisely the intersection of the solid S and the octant) to valid finite element topologies. The NIO cells that do not contain edges of bS are classified as Simple ("SNIO") and their finite element topologies are easily derived through template association.

For the NIO cells that contain edges and vertices of bS, decomposition through templates is not feasible because of the large number of possible configurations for the edge-cell intersection. These cells, labelled "Complex" NIO (CNIO), are decomposed through a set of element extractors that operate recursively on the topological and geometrical description of the cell. The starting point for recursive element extraction is a valid boundary representation of the polyhedron $\Re_c$, formed by the intersection of the CNIO cell and the cutting planes on bS. These operators are discussed in detail in the following section.

[1] Constructive Solid Geometry (CSG) exploits the notion of "adding" and "subtracting" simple building blocks ("primitives") via set-union and set-difference operations.

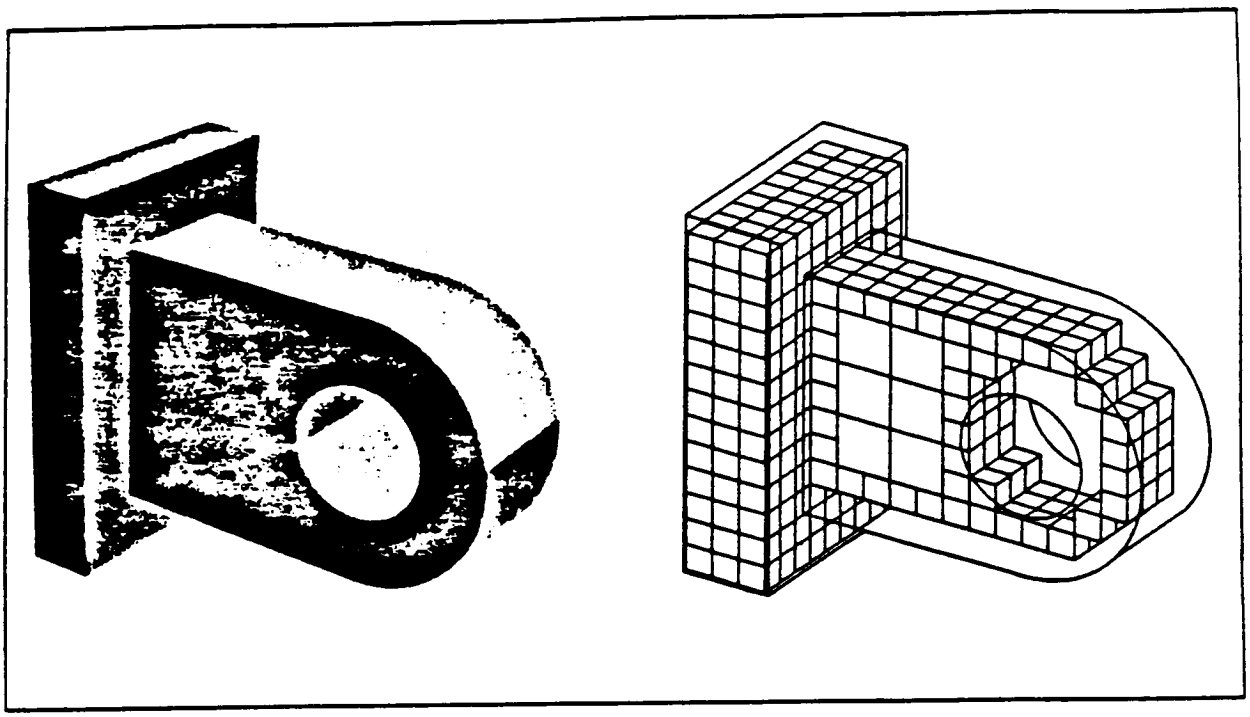

FIGURE 2.1 : A bracket and its interior octree.

The finite element mesh is complete at the end of second stage. The interior of the mesh consists of identical hexahedral elements and substructures associated with IN cells at resolution and higher levels, respectively. Also, the mesh inherits the hierarchical structure and the spatial addressability of the underlying octree decomposition.

As shown in [KELA86, 87], the regularity of the interior mesh together with the spatial addressability of the entire model provides the basis for a very powerful procedure for doing analysis as well as remeshing and reanalysis. Briefly, stiffness matrices are built and stored for all the non-OUT cells in the hierarchical tree (for identical interior elements and substructures they are copied into storage from precomputed values). This is done from the bottom up by assembling son matrices and condensing-out the interior degrees of freedom to build parent matrices at each level. A preliminary study on a 2-D implementation reported in [KELA87] suggests that this substructuring procedure is asymptotically more efficient than direct Gaussian reduction. For adaptive remeshing and reanalysis, spatial addressability allows efficient localized mesh modification. The reanalysis proceeds incrementally: the new stiffness matrices are inserted in the appropriate tree location and are combined with the stiffness of the unmodified elements and substructures.

In conclusion, the strict adherence of the FE mesh to the underlying octree structure offers some unique advantages for the analysis and, as such, is worth preserving. Therefore, stage 2 of the meshing procedure is designed in such a way as to leave intact the interior octree and the spatial addressability of the mesh.

2.1 Decomposition of 2-D NIO cells

The approach to 2-D NIO cells decomposition described in [KELA86, 87] is based on deriving finite element topologies exclusively through templates. In this case the number of required templates is small because of the following constraints imposed on the topology of the 2-D NIO cells:

(1) each NIO cell can be traversed by bS only once, such that

$$\text{NIO} \cap bS = \pi^1 \quad \text{(1-D simply connected polyhedra);} \tag{1}$$

(2) each NIO cell can contain at most one "vertex" node of bS;

(3) each edge of the NIO cell can have at most one intersection with bS.

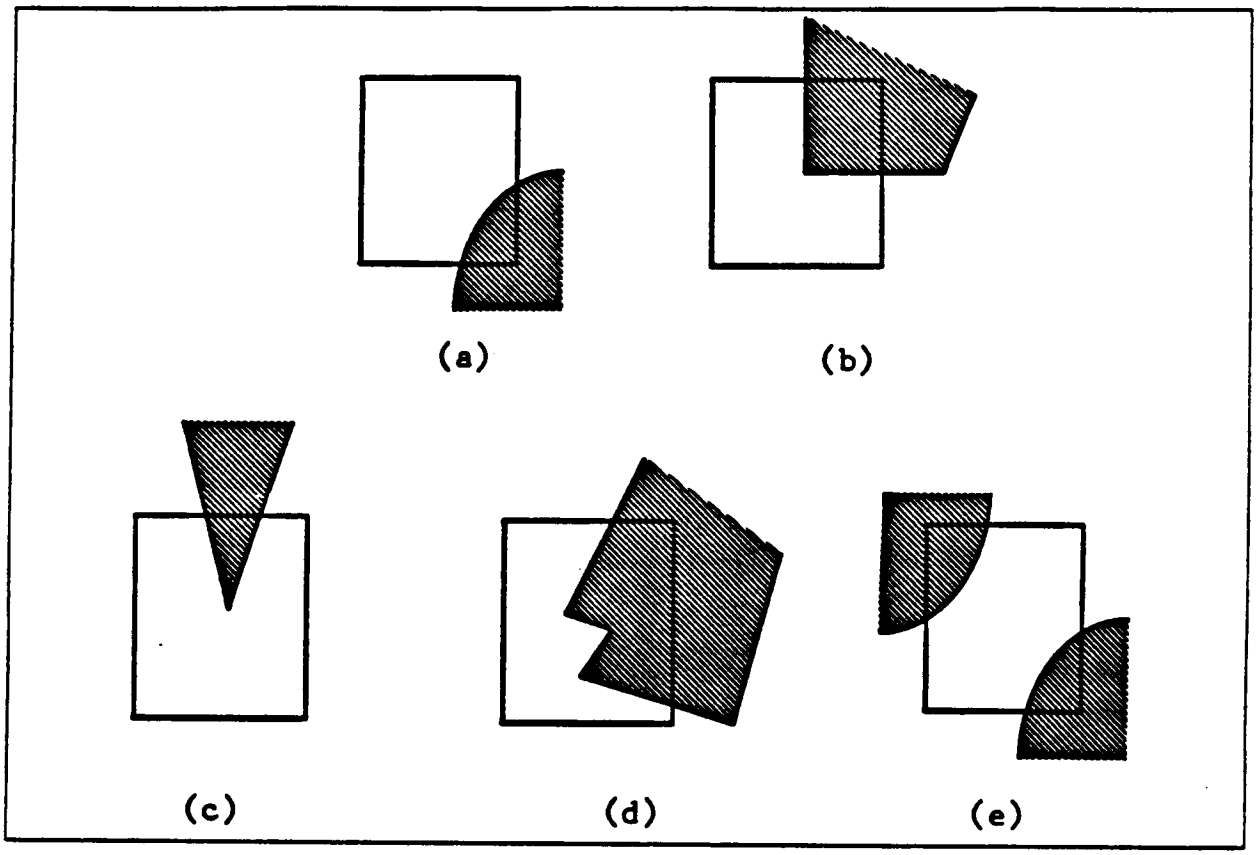

FIGURE 2.2: Valid (a,b) and invalid (c,d,e) 2-D NIO cells.

Valid and invalid 2-D NIO cells are shown in Figure 2.2. As shown in Figure 2.3, the derivation of element topologies, based on the above constraints, is simple. When the NIO cell does not contain any vertex, the element topology may be derived by traversing the boundary of the quadrant and counting the intersections with the object boundary. If the intersections are encountered on alternate edges, a quadrilateral element is mapped on to this cell. If the intersection takes place on adjacent edges, triangular elements are generated by connecting the intersection points to the appropriate cell node classified as IN.

For the case of NIO cells containing a vertex of bS, the vertex becomes a finite element node and triangles are generated by connecting this node to all the intersection points and the cell nodes that are inside the domain.

This simple decomposition approach – and the topological constraints on which it is based – cannot be extended to 3-D NIO cells because a 3-D bS contains edges. In this case, unless one imposes overly

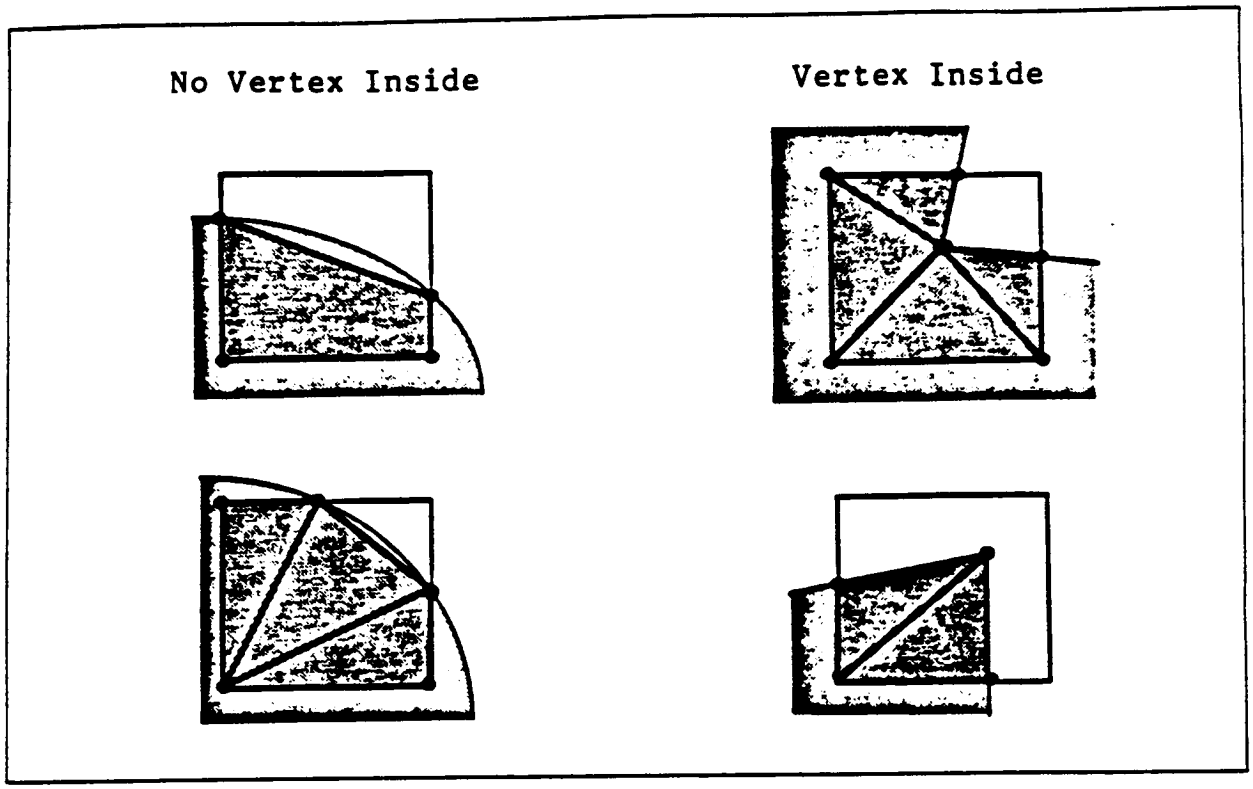

FIGURE 2.3 : Derivation of finite element topologies for 2-D NIO cells.

restrictive conditions on the way a bS edge is permitted to intersect a NIO cell, decomposition solely via template matching is infeasible.

An important property of the decomposition procedure described above is that each 2-D NIO cell contains all the topological information necessary to associate a finite element structure to the cell such that the continuity of the field variable across the cell boundary is ensured. Thus each 2-D NIO cell can be meshed independently from neighboring cells.

To prove this property we note that the interface between neighboring 2-D elements is an edge (1-D polyhedron). Therefore, to ensure continuity across the interface, the edge shared must be topologically identical, i.e., the edges must have the same bounding vertices (nodes) in both elements. Along the boundary of the NIO cell FE nodes are inserted only at the intersection points and at the cell vertices classified as IN. Because of this condition, any finite element topology introduced in the NIO cell contains only elements that have the correct interface with neighboring elements associated to either IN or NIO cells. Also, the insertion of triangular elements in a NIO cell does not disrupt the regularity of the mesh of square elements associated with the interior quadtree.

For 3-D problems, neighboring elements have a face in common (a 2-D polyhedron) and continuity requires that the shared face have the same set of bounding edges in both elements. In this case, the insertion of nodes on the NIO cell boundary only at the intersection points and at the cell vertices is not sufficient to ensure that 3-D NIO cell meshed independently will satisfy continuity across the interface. We shall expand on this problem later.

3 DECOMPOSITION OF 3-D NIO CELLS

The NIO cells are classified as SNIO or CNIO, based on the topological description of the associated polyhedron, $\Re$, defined as

$$\Re = \text{NIO} \cap^* S . \tag{2}$$

Here $\cap^*$ denotes a regularized intersection [REQU85]. If $\Re$ does not contain any vertex or edge of bS, the cell is classified as SNIO. In this case, $\Re_s$, the polyhedron associated with the SNIO cell, can be simply described as an octal cell in which a number of vertices have been shaved off by a single cutting surface, i.e.,

$$\Re_s = \textit{Octant} \oplus H_1 \tag{3}$$

where octant indicates an octal cell at resolution level, $\oplus$ a regularized boolean operation and H_1 is the cutting surface. Figure 3.1 shows a typical SNIO cell and its corresponding location in the solid part. Note that the associated polyhedron $\Re_s$ is a cube with four corners shaved off.

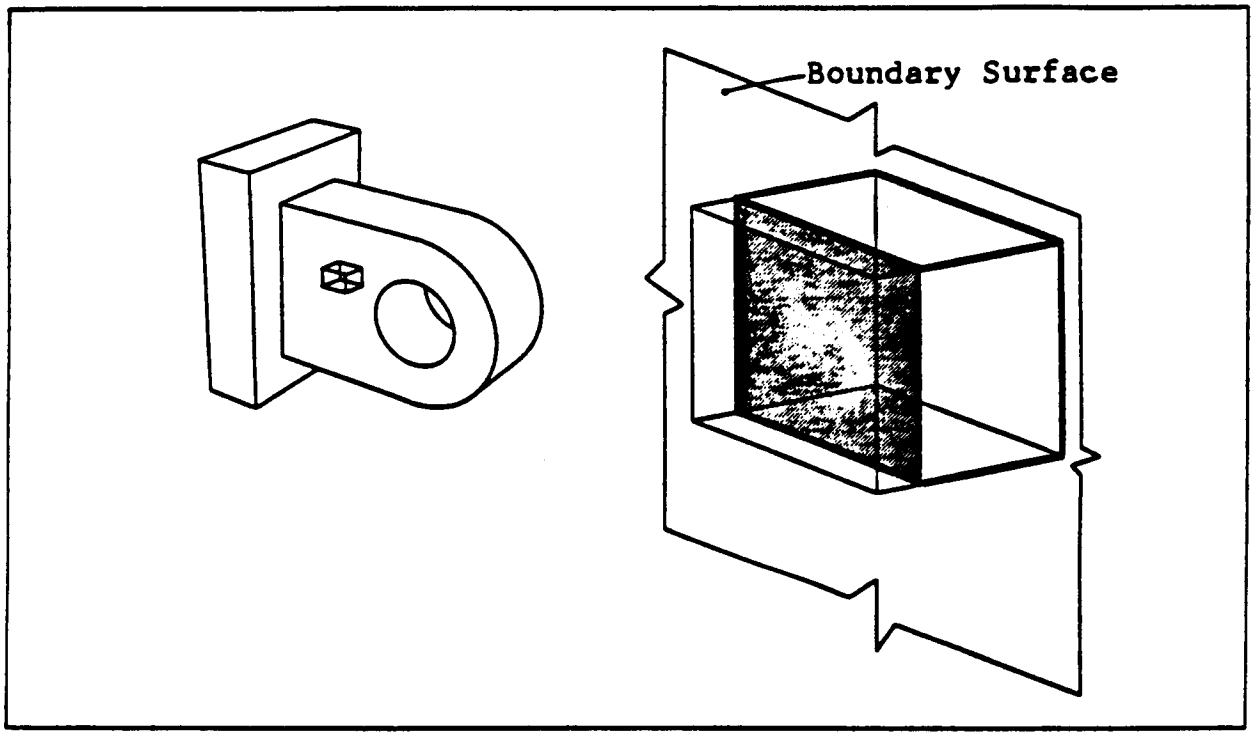

FIGURE 3.1 : A SNIO cell and its location on the solid part.

If $\Re$ contains vertices or edges of bS, the cell is classified as CNIO. Since a vertex is always defined by three or more intersecting surfaces and an edge by exactly two, the associated polyhedron can be represented as

$$\Re_c = \textit{Octant} \oplus H_1 \oplus H_2 \ldots \oplus H_n \tag{4}$$

where $H_1, H_2, \ldots H_n$ are cutting surfaces. Figure 3.2 shows a CNIO cell which contains three edges and a vertex of bS.

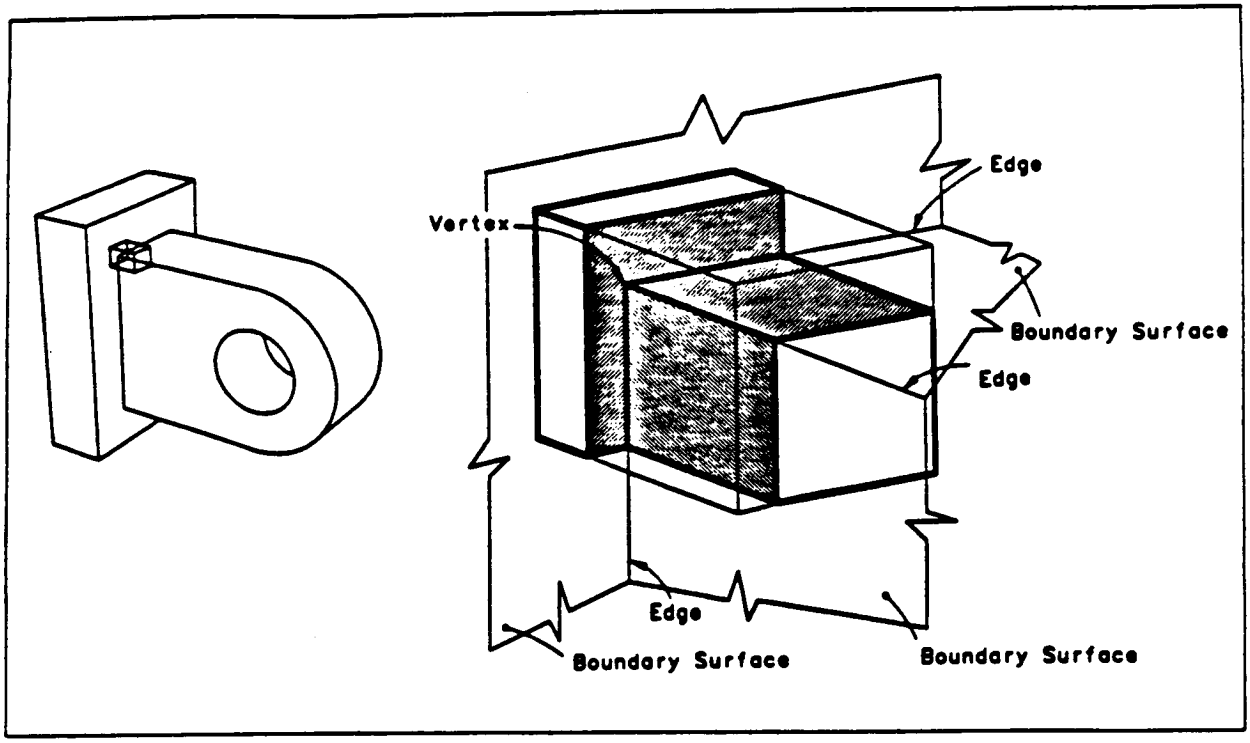

FIGURE 3.2: A CNIO cell and its location on the solid part.

3.1 Decomposition of SNIO cells

Since only seven different FE topologies are required for all possible SNIO cell configurations, the cell is decomposed by first selecting the appropriate template and then mapping the mesh from the template onto $\Re_s$. The template is chosen by counting the number of vertices shaved off by the cutting surface.

Figure 3.3 shows four cases of SNIO cells and the associated template derived meshes. The remaining three cases of possible SNIO cells, not illustrated in this figure, are the complements of (a), (b) and (c). The templates shown are based on linear hexahedral, pentahedral, wedge and tetrahedral elements[2]. The quadrilateral faces of the pentahedral and wedge elements mapped on to the square sides of the NIO cell ensure continuity along the interface between the NIO cell and the interior mesh. The union of the finite elements represents a planar approximation of the actual geometry with all the non-planar segments of δS intersected by the NIO cell replaced by triangular and quadrilateral bilinear patches. Finally, we note that most of the NIO cells are classified as SNIO cells and their decomposition through template matching is computationally inexpensive.

3.2 Decomposition of CNIO cells

The element "extractors", shown in Figure 3.4, are a modified version of the operators presented in [WOO84]. The operators τ_1 and τ_2 produce tetrahedral elements while τ_3 extracts pentahedra based on the square faces of $\Re_c$ that correspond to the original faces of CNIO cell. These operators work as follows.

[2] Linear pentahedral, wedge and tetrahedral elements can be generated by collapsing a standard isoparametric brick element [BATH82].

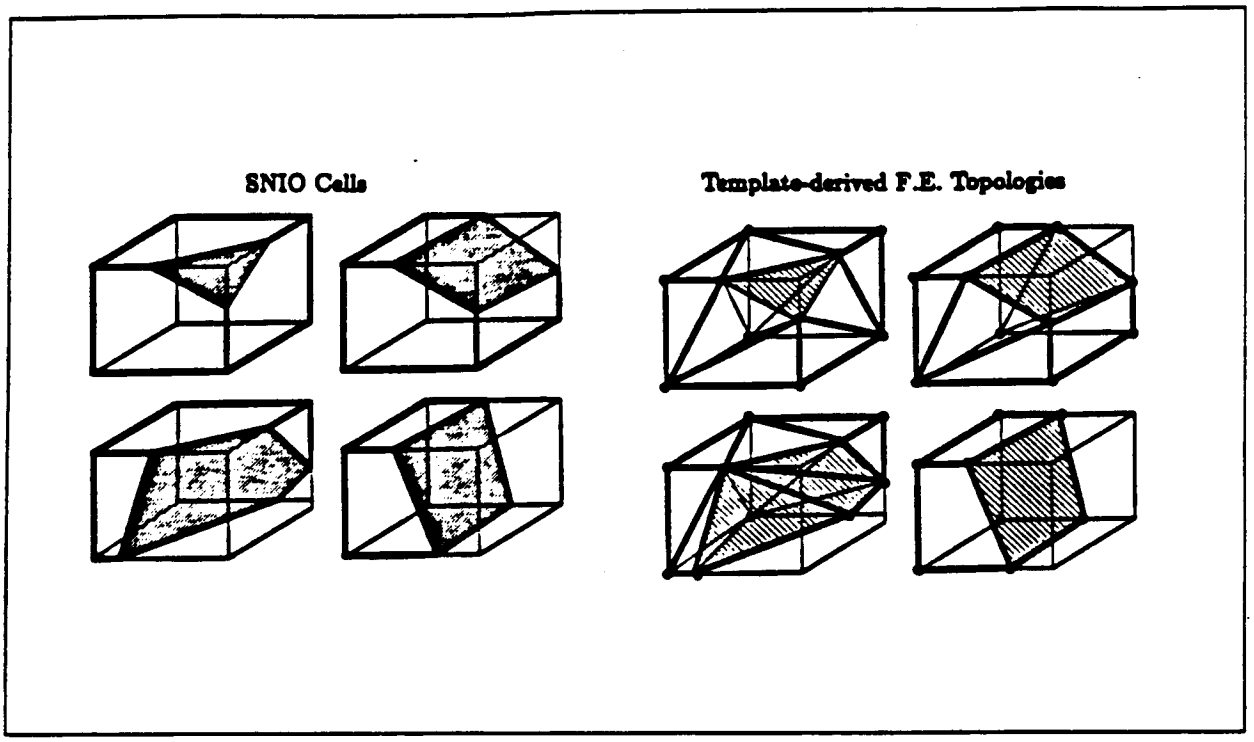

FIGURE 3.3 : SNIO cells and associated templates.

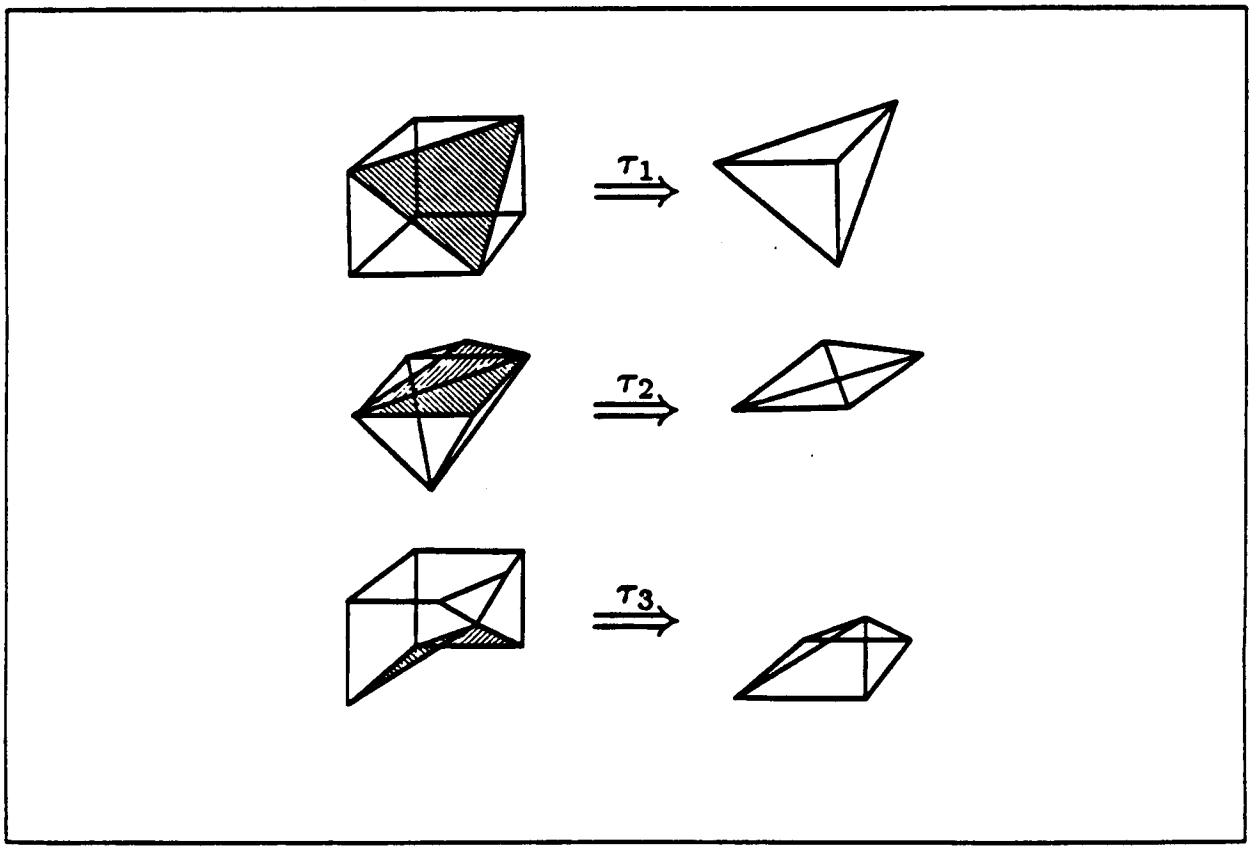

FIGURE 3.4 : Element extractors for CNIO cells.

1) τ_1 scans the boundary representation of $\Re_e$ searching for convex trivalent vertices. When such a vertex is found τ_1 extracts a tetrahedron by introducing a single cut in the domain (this corresponds to the "slicing" operation in [WOO84]).

2) τ_2 is applied when all the convex trivalent vertices are exhausted. This operator uses a convex edge to extract a tetrahedron by introducing two cuts in the domain – referred to as "digging" into the domain in [WOO84].

3) τ_3 looks for faces of $\Re_e$ that correspond to original cell faces and extracts a pentahedron by introducing multiple cuts that vary according to the location of the apex vertex. The choice of the apex vertex is based on interference considerations. The operator τ_3 is applied before τ_1 and τ_2 in order to preserve all the original cell faces contained in $\Re_e$.

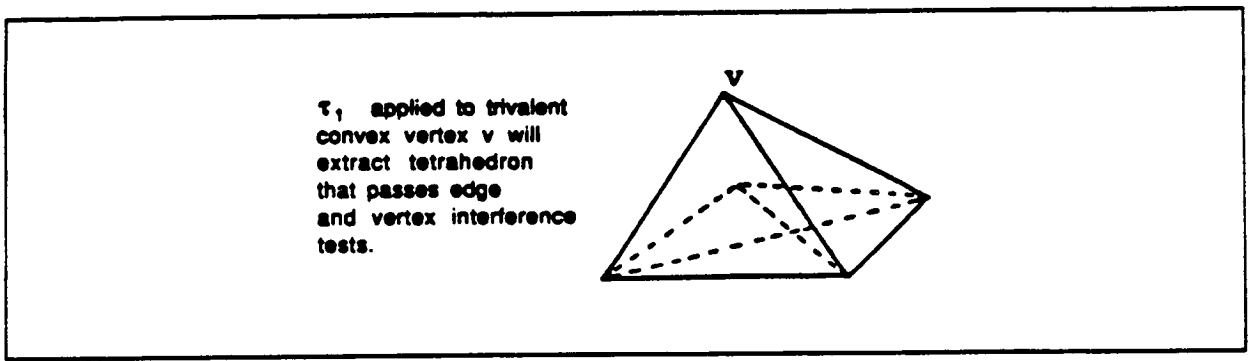

FIGURE 3.5 : Pathological case for the interference test in [WOO84].

Before each extraction the validity of the candidate tetrahedron or pentahedron is checked through a series of tests. As in [WOO84], the vertices and edges of $\Re_e$ are checked for interference with the faces of the candidate element. More precisely, the interference test ensures that: (i) no vertex of $\Re_e$ lies on any of the faces of the candidate element, and (ii) no edge of $\Re_e$ intersects any of the faces of the candidate element. This test is not enough, however, to ensure the validity of the element – see the exemplary pathological case illustrated in Figure 3.5. To remove the ambiguity, an additional check is performed by classifying the centroid of each face of the candidate element against $\Re_e$. If all centroids are classified as ON or IN the element is valid.

The implementation of the element extractors and the geometric checks described above requires a point-membership classifier (PMC) — a function that returns the classification of a point p with respect to the polyhedron $\Re_e$ as

$$PMC(p, \Re_e) \;=\; (In, On, Out)\,. \tag{5}$$

The PMC developed for the present work operates on planar polyhedra and is based on ray casting algorithms [KALA82].

The boundary representation (BRep) structure used for maintaining and updating the topology of $\Re_e$ has two graphs imbedded in it: (i) Face → Edge → Vertex and (ii) Vertex → Edge → Face. This double structure provides greater flexibility while manipulating the BRep for the polyhedron, because it reduces the number of scans required to retrieve the necessary information about the boundary. The PMC permits the classification of the edges and the vertices in the BRep as convex or concave. This piece of information is crucial for element extraction and must be updated after each element removal.

All the operators used for element extraction preserve the differential form of the Euler-Poincare formula [BAUM72]

$$\nabla F - \nabla E + \nabla V \;=\; 0 \qquad (6)$$

where F is the number of faces, E is the number of edges and V the number of vertices in the polyhedron. Provided that the initial polyhedron is valid, the satisfaction of equation (6) ensures that the validity is maintained at each stage of the extraction.

Figure 3.6 shows different stages of the element extraction on the $\Re_e$ associated with the CNIO cell in Fig. 3.2. The operator τ_3 is applied to extract a pentahedral element whose base is the original cell face. This is followed by the recursive application of the operator τ_1 until all trivalent convex vertices are exhausted. Operator τ_2 takes over until one or more trivalent convex vertices become available and τ_1 can be applied again. This progressively reduces the domain to a single tetrahedron.

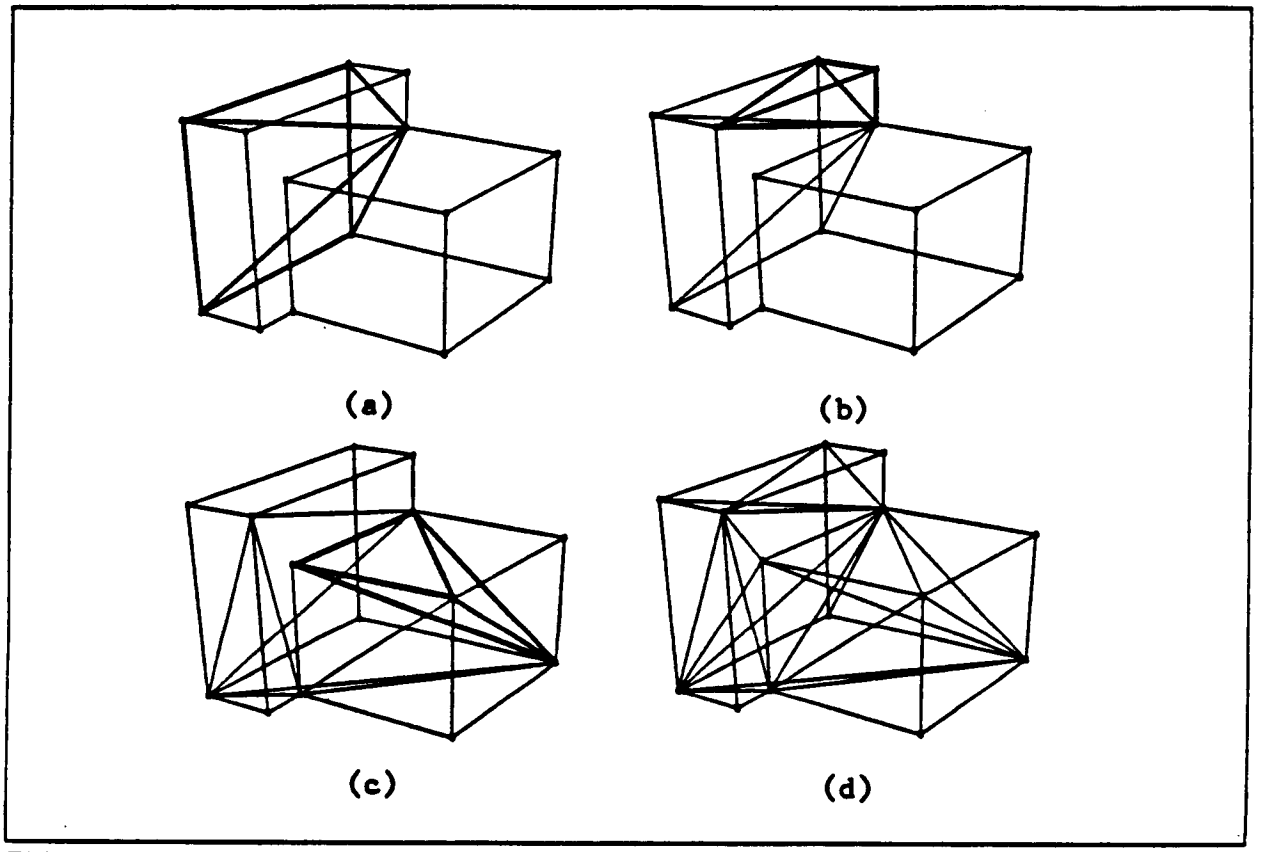

FIGURE 3.6 : Element extraction on a CNIO cell: τ_3 extraction of a pentahedron (a), a tetrahedron extracted via τ_1 (b) and τ_2 (c), the complete element topology.

The computational cost for deriving the BRep of $\Re_e$ and decomposing it through element extraction is considerably higher than that associated with SNIO cell decomposition. We note, however, that the number of CNIO cells is relatively small.

4 BOUNDARY EVALUATION FOR CNIO CELLS

In the preceding section we indicated that a boundary representation of the polyhedron $\Re_e$ is needed for the element extraction. The standard approach to derive the boundary for solids described in a CSG environment is to intersect the faces of all the primitives that constitute the CSG definition of the object and classify the resulting edges against the combinatorial tree (this operation is called boundary merging [REQU85]). Since this merging process involves all the primitives, boundary evaluation for a CSG described solid is in general a computationally expensive procedure[3].

The polyhedron $\Re_e$ is formed by the intersection of the CNIO octant and the original solid, i.e.,

$$\Re_e = CNIO \cap^* S \tag{7}$$

and therefore its boundary evaluation appears to require the boundary merging of the complete solid S and the CNIO cell. We note, however, that generally the cell under consideration is spatially localized, i.e., each CNIO cell intersects only a limited number of primitives. In this case the boundary of $\Re_e$ can be obtained by merging only the boundary of the primitives which interfere with the cell.

The primitive incidence information required to generate the necessary set of tentative edges is produced the following way. At the last level of the octree decomposition every NIO cell is classified against each of the primitives in the CSG tree. When the cell is classified "ON" a primitive, the primitive is added to the incidence information carried with the cell. Hence, at the end of the classification, each NIO cell points to the subset of the CSG primitives that "interact" spatially with the cell. As indicated in [TILO81], primitive incidence leads to "pruning" of the CSG tree and, consequently, reduces the computational cost of boundary merging.

The boundary of $\Re_e$ is necessarily contained in the boundary of the CNIO cell and of the primitives incident upon the cell. Therefore the tentative set of edges generated merging the incident primitives and the cell suffices for building the boundary representation of $\Re_e$. Since several CNIO cells may be incident upon a small number of primitives, exploiting primitive incidence may save considerable computational time during mesh generation. Fig. 4.1 illustrates tree pruning for the CNIO cell shown in Fig. 3.2.

5 DISCUSSION

As indicated in the previous sections, by adhereing to the underlying octal cell decomposition the mesh acquires hierarchical structure, spatial addressability and interior geometrical regularity. We consider these three properties central to the automation of finite element analysis. Therefore our approach to octree based automatic meshing is focussed on preserving a tight correspondence between the finite element and the cell structure. In particular, this requires embedding a finite element topology in the NIO cells without disrupting the global structure and addressability of the mesh as well as the regularity of the interior octree.

We treat NIO cells in a selective way: element extractors are used only for those – relatively few – cases for which template matching is infeasible. Template controlled decompostion is appealing because it is

[3] The asymptotic time complexity for boundary evaluation ranges between $O(n^3 \log n)$ and $O(n^4)$, where n is the number of primitives [TILO81].

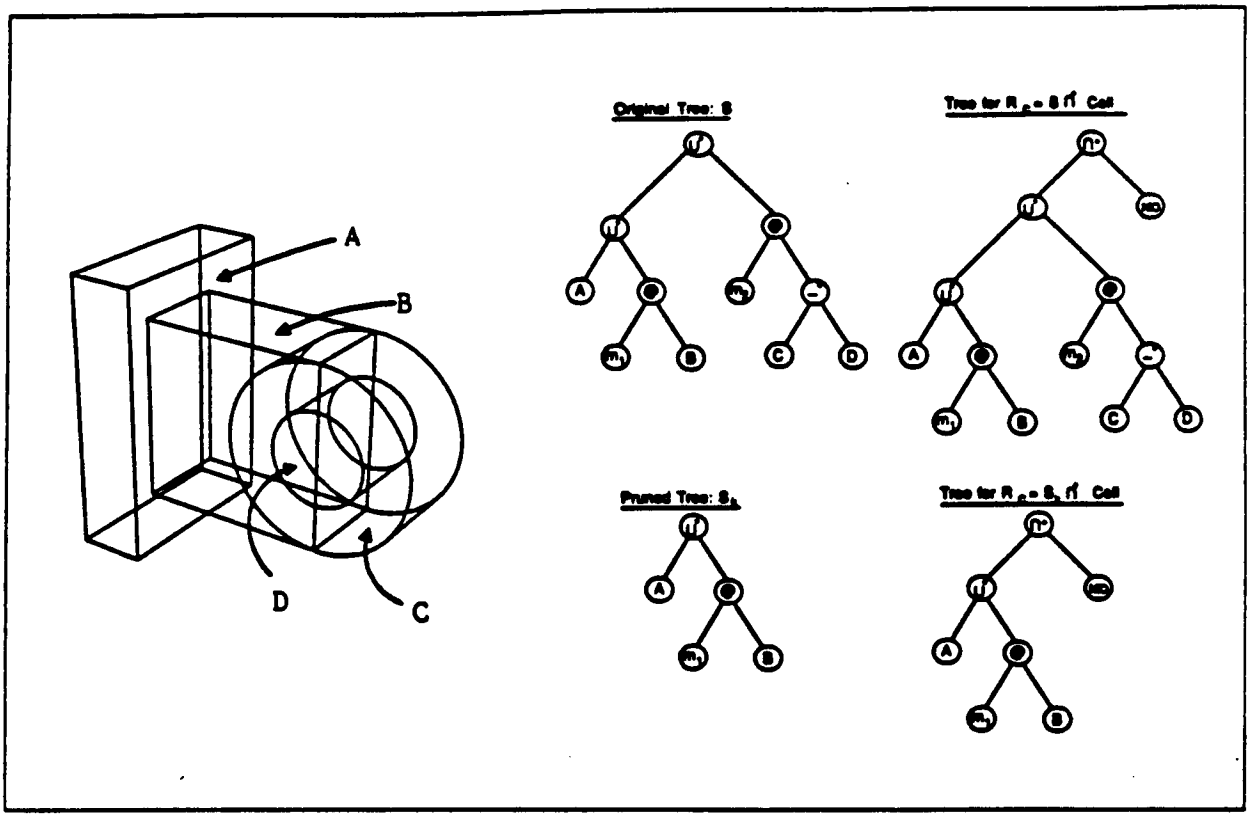

FIGURE 4.1 : Tree pruning for a CNIO cell. (A, B, C, D are primitives; m_1, m_2 are motions; $\cup^*, \cap^*, -^*$ represent regularized Boolean operations; @ indicates the application of a motion.

computationally inexpensive and allows a good degree of control on the elements which are inserted in the NIO cell. Conversely, element extractors require building and maintaining a sophisticated data structure and provide a very limited amount of control on the mesh. The exclusive use of mapping or element extraction on all the boundary cells – as proposed in [YERR84] and [YERR85], respectively – is either too limited for handling complex geometries (the former) or computationally too demanding for practical implementation (the latter). The selective use of the two algorithms based on the preliminary NIO cell classification described in this paper results in a flexible approach designed to exploit the advantages of both types of decomposition.

The algorithms discussed here are currently being implemented in an experimental code built on the PADL-2 modelling system. PADL-2 provides the utilities for modelling the solid and extracting the octree. Also, the geometric routines contained in the modeller are used extensively to perform the operations required for the SNIO/CNIO cell classification, the SNIO template mapping and the derivation of the CNIO boundary representation. In particular, boundary evaluation is done by first pruning the CSG tree and then using the PADL-2 incremental boundary evaluator [HART85]. The CNIO cell decomposition is carried out in an independent modelling environment based on the BRep structure described in Section 3. The implementation of the element extractors is built on a specialized point-membership classifier that operates on planar polyhedra.

To complete the implementation of our meshing algorithm we have to resolve some specific issues related to interfacing CNIO cells with IN and SNIO cells. Our plans are the following. The CNIO/IN interface is generally taken care of by using pentahedral elements. Whenever that is not possible, the propagation of triangular faces is contained within the adjacent IN cell with a two-step procedure: (i) decompose

the IN cell into 6 identical pyramids with the apex at the cell centriod and (ii) split the pyramid on the interface into two tetrahedral elements. The interface between SNIO/CNIO cells can be modelled by either embedding the edges on the SNIO face into the BRep associated with the CNIO cell, or by modifying locally the SNIO mapped mesh to reflect the entities on the CNIO face. Note that the task of identifying the two cells sharing a face is considerably simplified because of the spatial addressability of the cell (and element) structure.

In conclusion, the approach presented here resolves efficiently the geometrical and topological issues related to octree based automatic meshing and – in analogy with the quadtree structures described in [KELA86] – opens a promising avenue for self-adaptive analysis.

ACKNOWLEDGEMENTS

Herbert Voelcker of Cornell University, Ajay Kela of General Electric Company, and Aristides Requicha of the University of Southern California contributed to this research. The figures were produced on equipment donated by Tektronix, Inc. Other Industrial Associate companies of the Production Automation Project provided sustaining support. The findings and opinions expressed here do not reflect the views of the sponsors.

REFERENCES

[BATH82] K. J. Bathe, *Finite Element Procedures in Engineering Analysis*. New Jersey: Prentice-Hall, 1982.

[BAUM72] B. G. Baumgart, "Winged edge polyhedran representation", STAN-CS-320, Stanford Artificial Intelligence Project, Stanford University, October 1972.

[CAVE85] J. C. Cavendish, D. A. Field and W. H. Frey, "An approach to automatic three-dimensional finite element mesh generation", *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 329-347, 1985.

[HART83] E. E. Hartquist, "Public PADL-2", *IEEE Computer Graphics and Applications*, vol. 3, no. 7, pp. 30-31, October 1983.

[HART85] E. E. Hartquist, "PP2/2.N Boundary Evaluator", Incremental Boundary Evaluator Doc. No. 4, Production Automation Project, University of Rochester, December 1985.

[JACK80] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects", *Computer Graphics & Image Processing*, vol. 4, no. 3, pp. 249-270, November 1980.

[KALA82] Y. E. Kalay, "Determining the spatial containment of a point in general polyhedra", *Computer Graphics & Image Processing*, vol. 19, no. 4, pp. 303-334, August 1982.

[KELA86] A. Kela, R. Perucchio and H. B. Voelcker, "Toward automatic finite element analysis", *ASME Computers in Mechanical Engineering*, vol. 5, no. 1, pp. 57-71, July 1986.

[KELA87] A. Kela, "Automatic finite element mesh generation and self-adaptive incremental analysis through solid modeling", Ph.D. Dissertation, Dept. of Mechanical Engineering, University of Rochester, January 1987.

[LEE82] Y. T. Lee and A. A. G. Requicha, "Algorithms for computing the volume and other integral properties of solids: Part II - A family of algorithms based on representation conversion and cellular approximation", *Communications of the ACM*, vol. 25, no. 9, pp. 642-650, September 1982.

[REQU85] A. A. G. Requicha and H. B. Voelcker, "Boolean operations in solid modelling: Boundary evaluation and merging algorithms", *Proceedings of the IEEE*, vol. 3, no. 1, pp. 30-44, January 1985.

[SHEP85] M. S. Shephard, "Finite element modeling within an integrated geometric modeling environment: Part I - Mesh generation", *Engineering with Computers*, vol. 1, pp. 61-71, 1985.

[TILO81] R. B. Tilove, "Line/polygon classification: A study of the complexity of geometric computation", *IEEE Computer Graphics and Applications*, vol. 1, no. 2, pp. 75-88, April 1981.

[WOO84] T. C. Woo and T. Thomasma, "An algorithm for generating solid elements in objects with holes", *Computers & Structures*, vol. 18, no. 2, pp. 333-342, 1984.

[WÖRD84] B. Wördenweber, "Finite-element analysis for the naive user", in M. S. Pickett and J. W. Boyse, Eds., *Solid Modelling by Computers*. New York: Plenum Press, 1984, pp. 81-102.

[YERR84] M. A. Yerry and M. S. Shephard, "Automatic three–dimensional mesh generation by the modified–octree technique", *International Journal for Numerical Methods in Engineering*, vol. 20, pp. 1965–1990, 1984.

[YERR85] M. A. Yerry and M. S. Shephard, "Trends in engineering software and hardware – Automatic mesh generation for three-dimensional solids", *Computers & Structures*, vol. 20, no. 1-3, pp. 31-39, 1985.